INFOGR – Computer Graphics

Jacco Bikker & Debabrata Panja - April-July 2017

Lecture 3: “Ray Tracing”

Welcome!

Today’s Agenda:

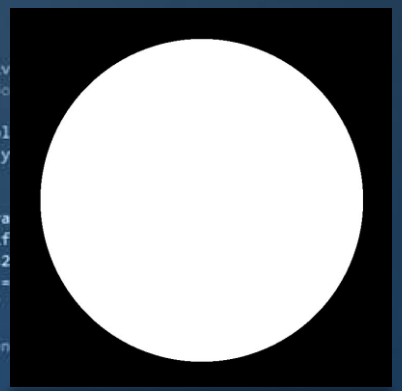
- Ray Tracing
- Intersections
- Shading
- Assignment 2
- Textures

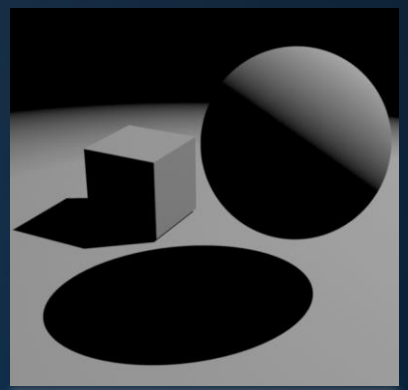
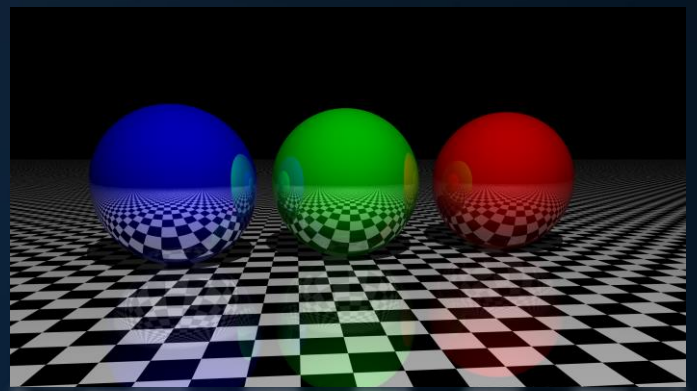
Ray Tracing

PART 1: Introduction & shading (today)

PART 2: Reflections, refraction, absorption (next week)

PART 3: Path Tracing (later)



Ray Tracing

Ray Tracing:

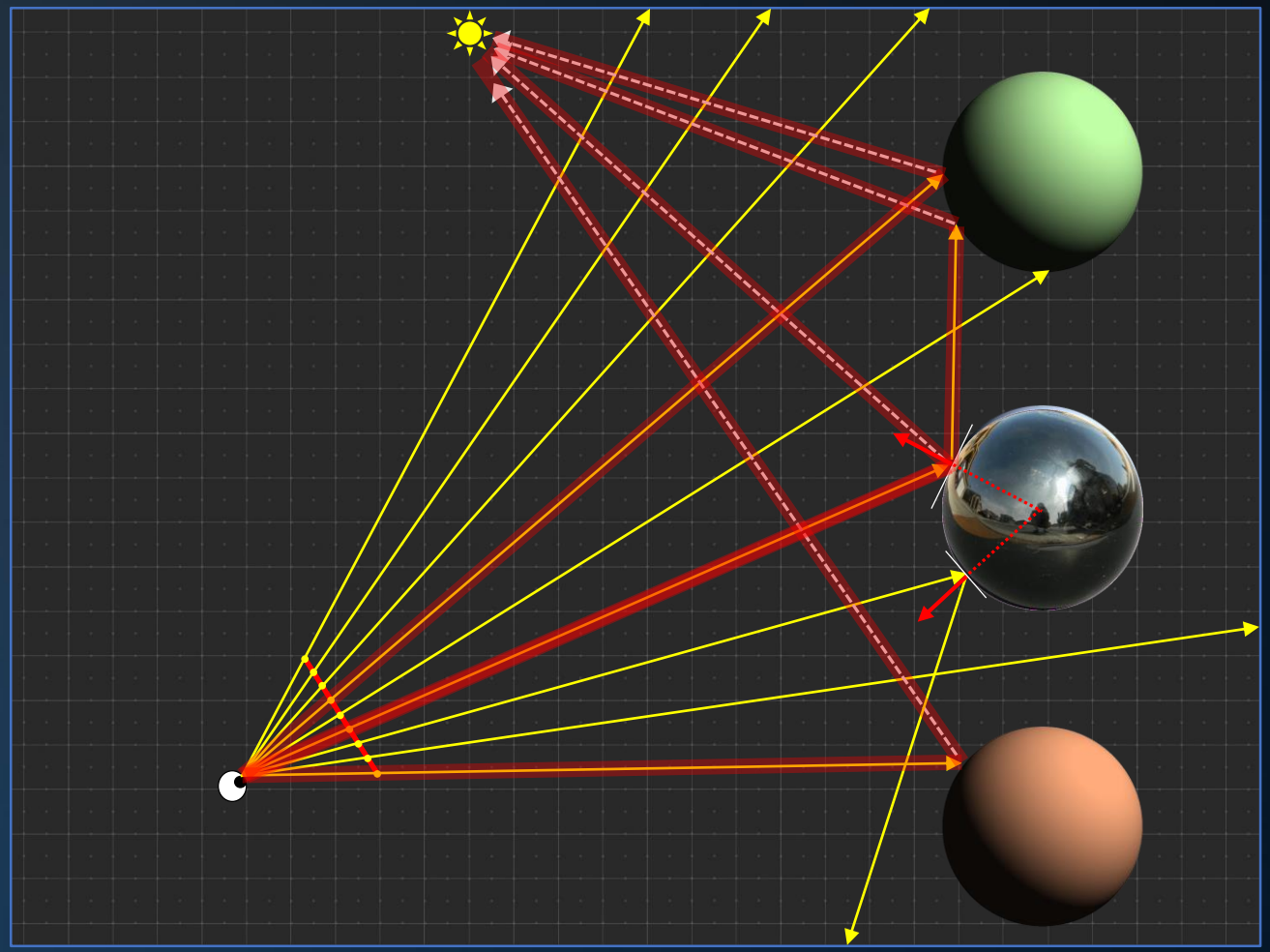
World space

- Geometry
- Eye
- Screen plane
- Screen pixels
- Primary rays
- Intersections
- Point light
- Shadow rays

Light transport

- Extension rays

Light transport



Ray Tracing

Ray Tracing:

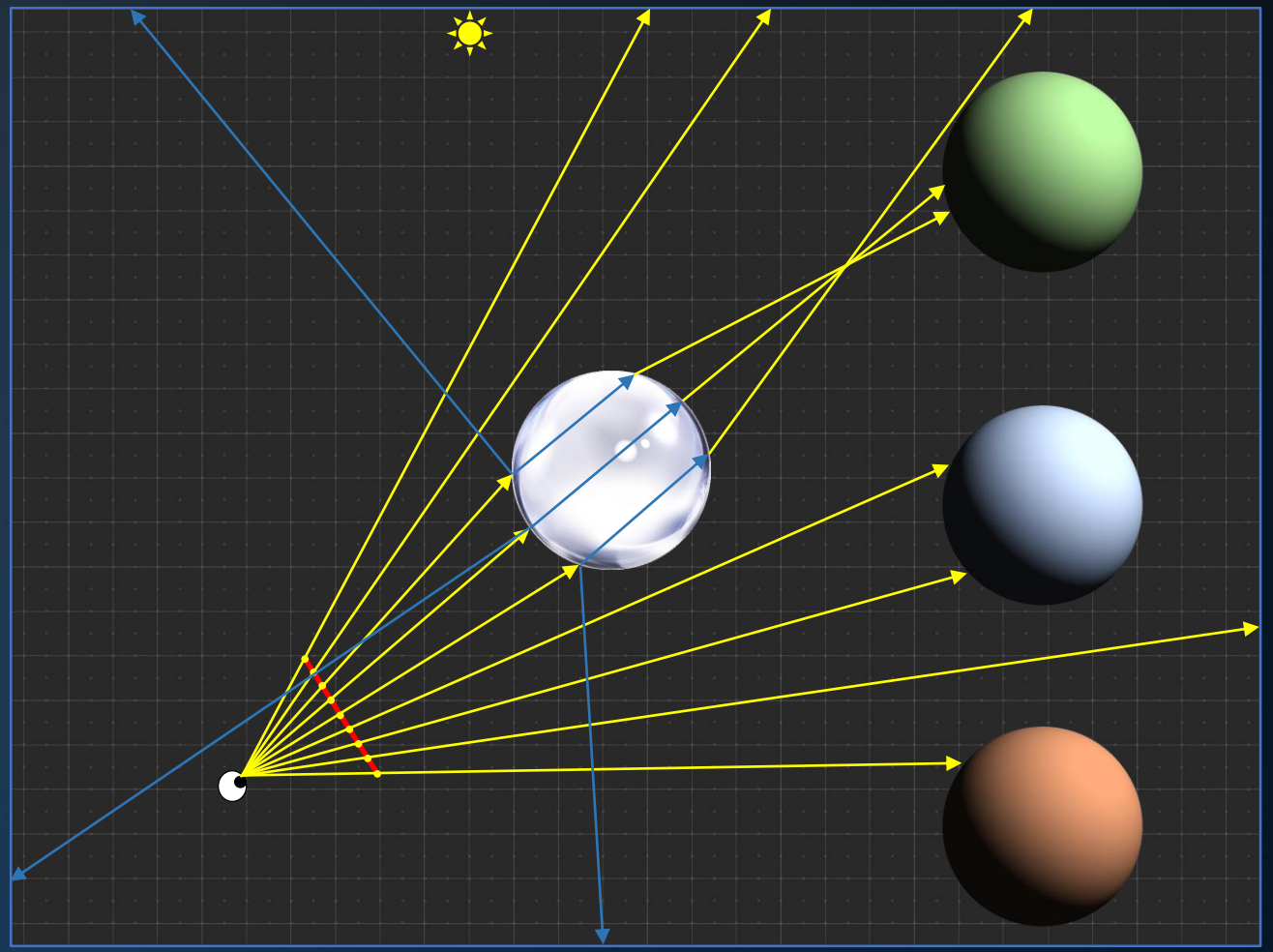
World space

- Geometry
- Eye
- Screen plane
- Screen pixels
- Primary rays
- Intersections
- Point light
- Shadow rays

Light transport

- Extension rays

Light transport



Ray Tracing

Ray Tracing:

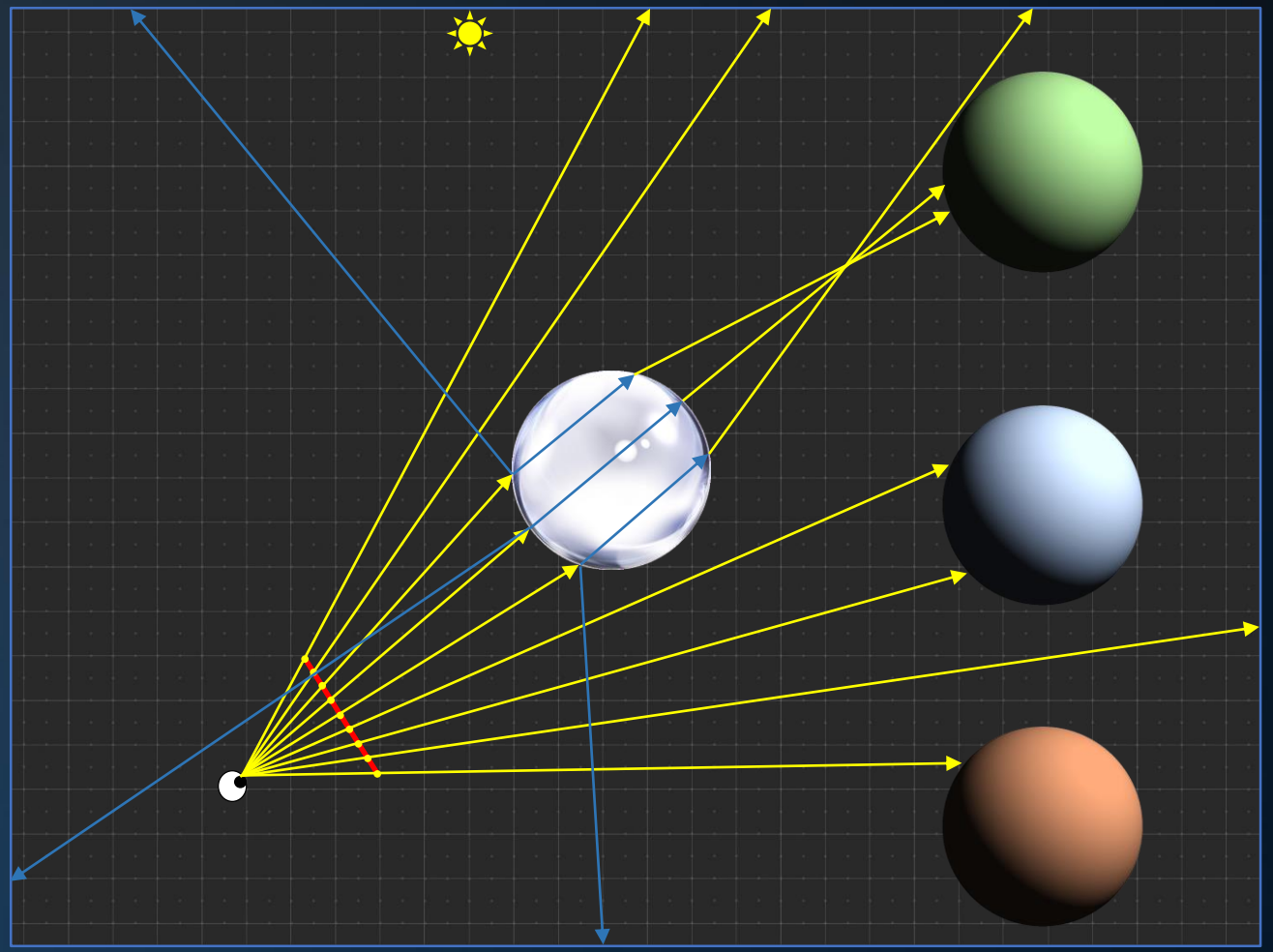
World space

- Geometry
- Eye
- Screen plane
- Screen pixels
- Primary rays
- Intersections
- Point light
- Shadow rays

Light transport

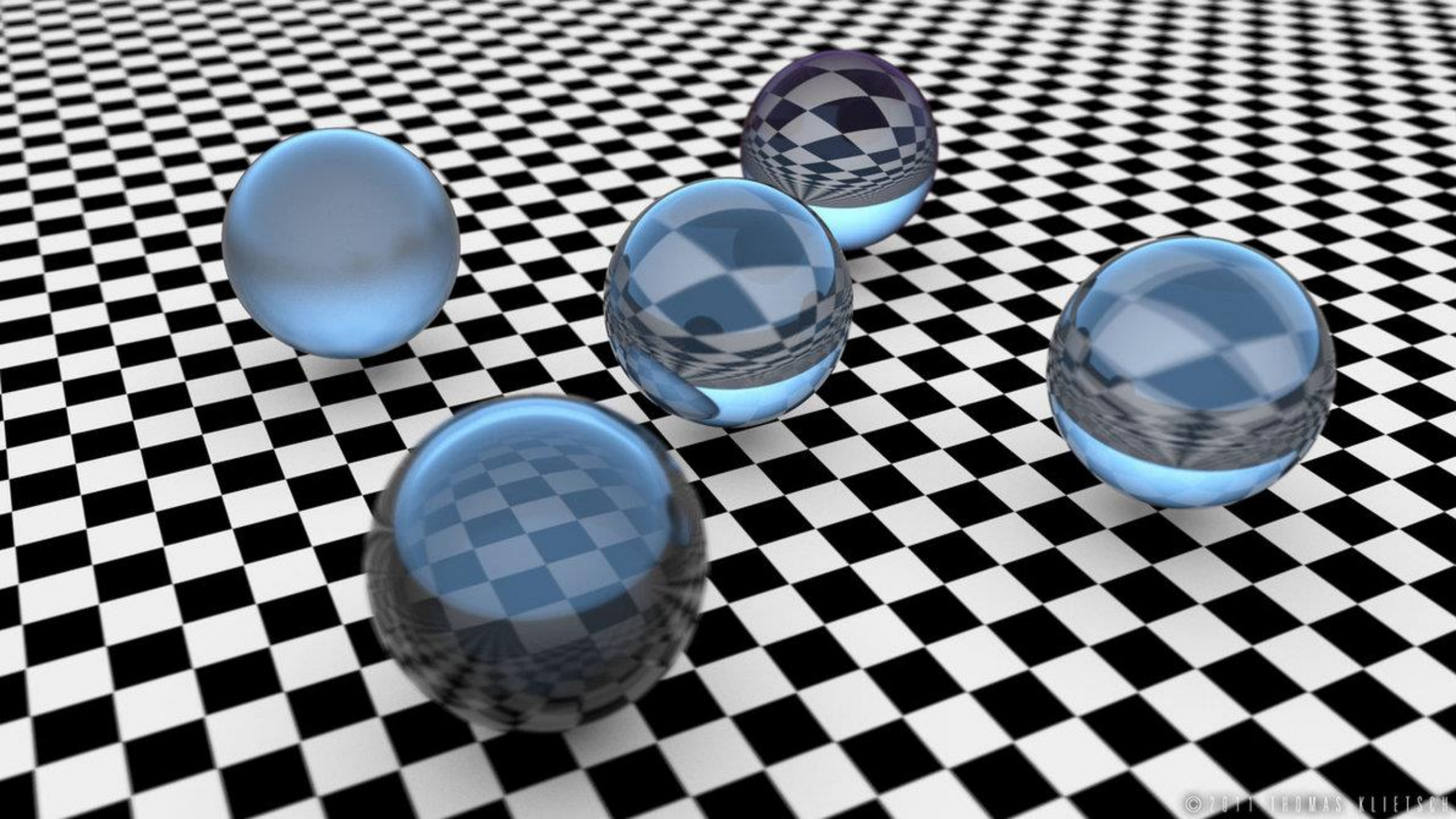
- Extension rays

Light transport

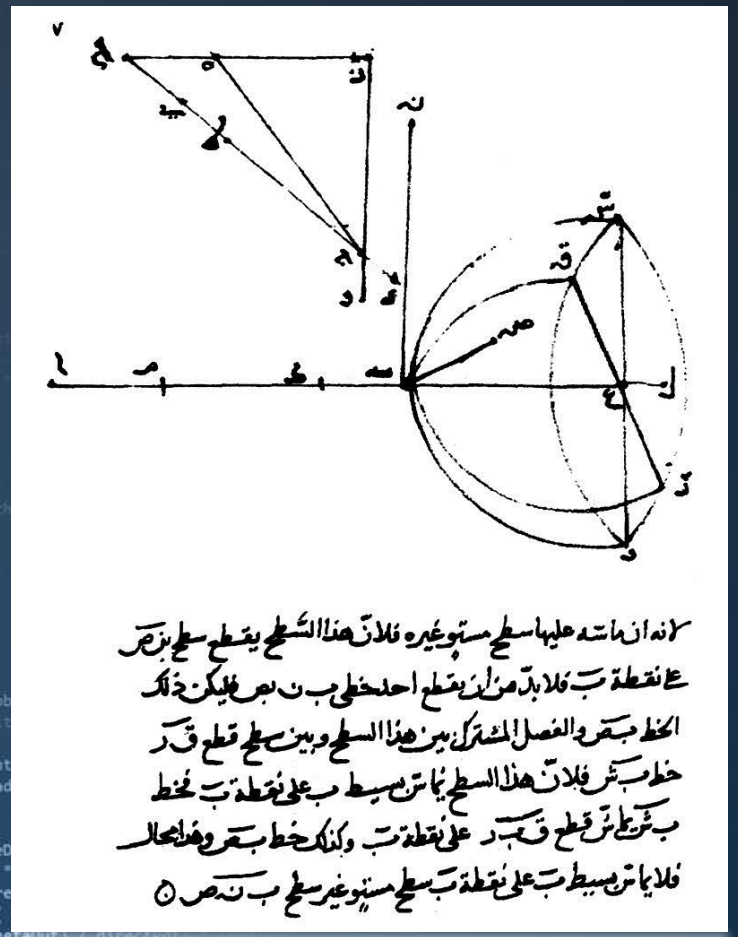


Note:

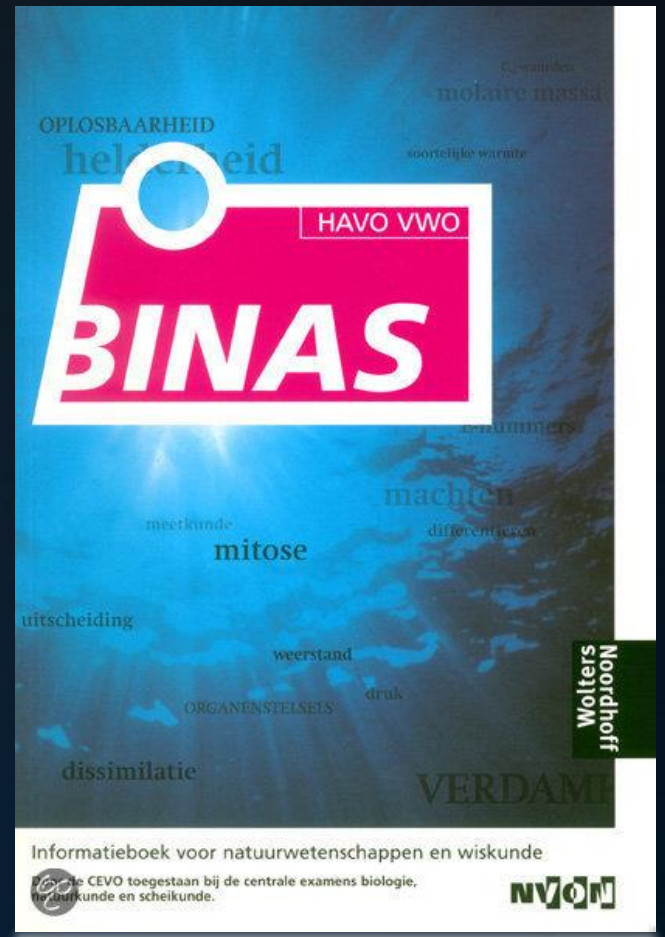
We are calculating light transport *backwards.*


© 2011 THOMAS KLIETSCH

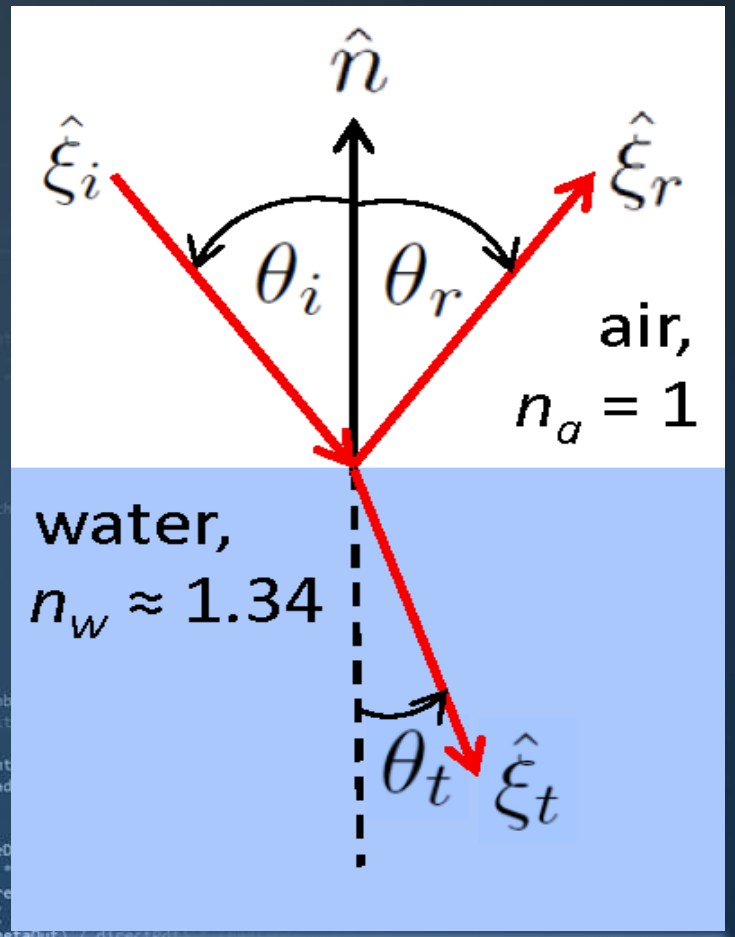
Ray Tracing

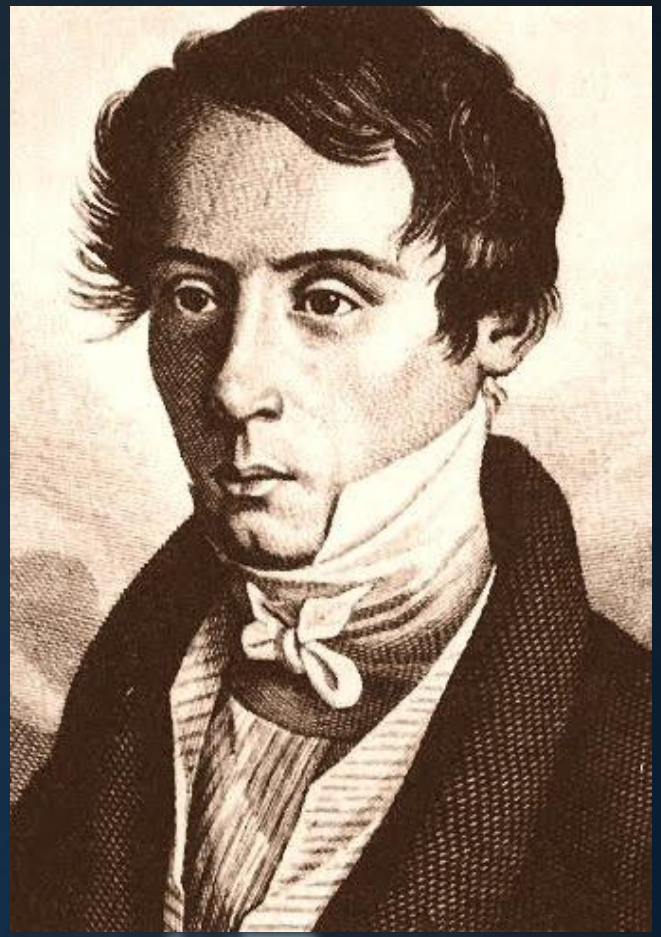


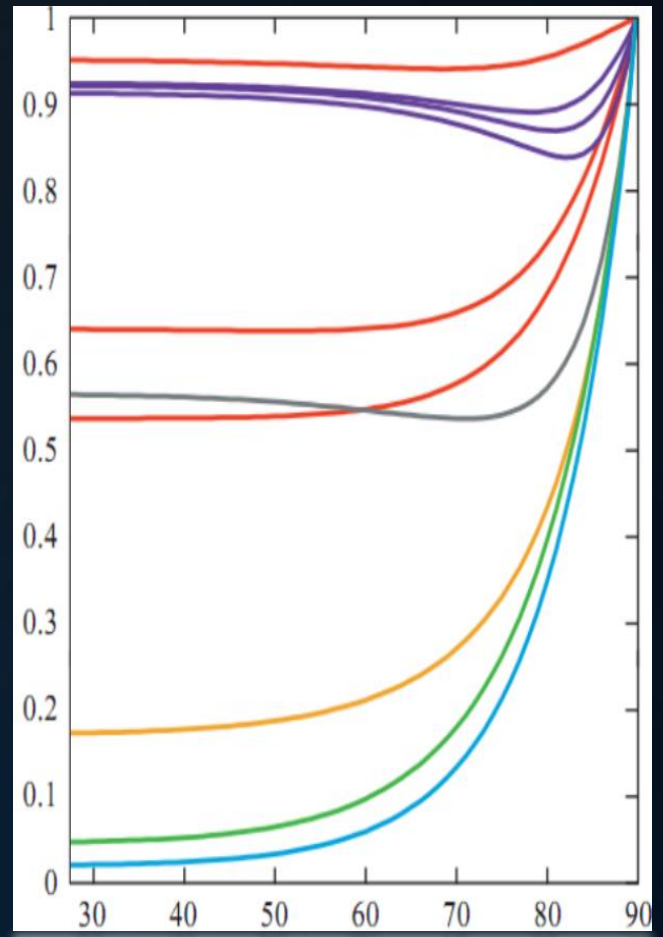

WILLEBRORDUS SNELLIUS.
Mathescos Professor.


OPLOSBAARHEID
molaire massa
soortelijke warmte
HAVO VWO
BINAS
machten
meetkunde
differentiëren
mitose
uitscheiding
weerstand
druk
ORGANENSTELSELS
dissimilatie
VERDAMPING
Wolters Noordhoff
Informatieboek voor natuurwetenschappen en wiskunde
NVON

Ray Tracing


$\hat{n}$
$\hat{\xi}_i$
$\hat{\xi}_r$
θ_i
θ_r
air,
$n_a = 1$
water,
$n_w \approx 1.34$
θ_t
$\hat{\xi}_t$




1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
30
40
50
60
70
80
90

Ray Tracing

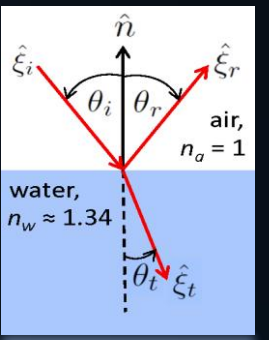
Physical basis

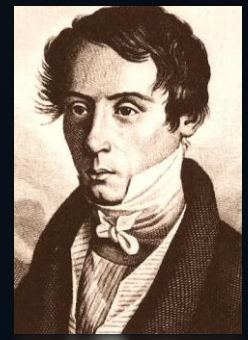
Ray tracing uses *ray optics* to simulate the behavior of light in a virtual environment.

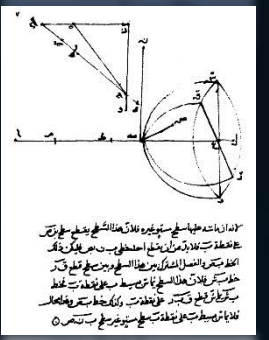
It does so by finding light transport paths:

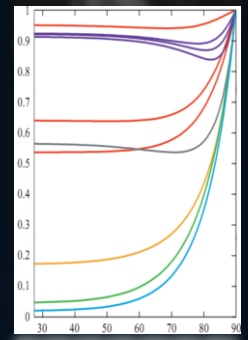
- From the ‘eye’
- Through a pixel
- Via scene surfaces
- To one or more light sources.

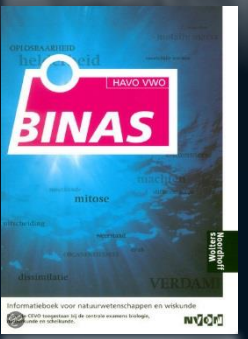
At each surface, the light is modulated.
The final value is deposited at the pixel
(simulating reception by a sensor).

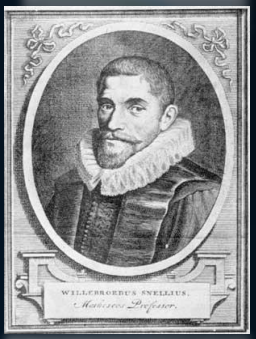












T. Whitted, “An Improved Illumination Model for Shaded Display”, ACM, 1980.

Today’s Agenda:

- Ray Tracing
- Intersections
- Shading
- Assignment 2
- Textures

Intersections

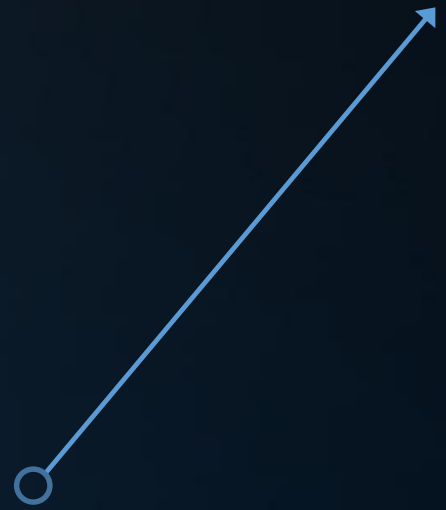
Ray definition

A ray is an infinite line with a start point:

$p(t) = O + t\vec{D}$, where $t > 0$.

```
struct Ray
{
    float3 O;    // ray origin
    float3 D;    // ray direction
    float t;     // distance
};
```

The ray direction is generally *normalized*.



Intersect

Ray setup

A ray is initially shot through a pixel on the screen plane.
The screen plane is defined in world space:

Only if $\vec{V} = (0,0,1)$ *of course.*

Camera position: $\mathrm{E} = (0,0,0)$
View direction: $\vec{V}$
Screen center: $\mathrm{C} = E + d\vec{V}$
Screen corners: $\mathrm{p}_0 = C + (-1,-1,0)$, $p_1 = C + (1,-1,0)$, $p_2 = C + (-1,1,0)$

From here:

- Change FOV by altering d;
- Transform camera by multiplying $\mathrm{E}, p_0, p_1, p_2$ with the camera matrix.

Intersect

Ray setup

Point on the screen:

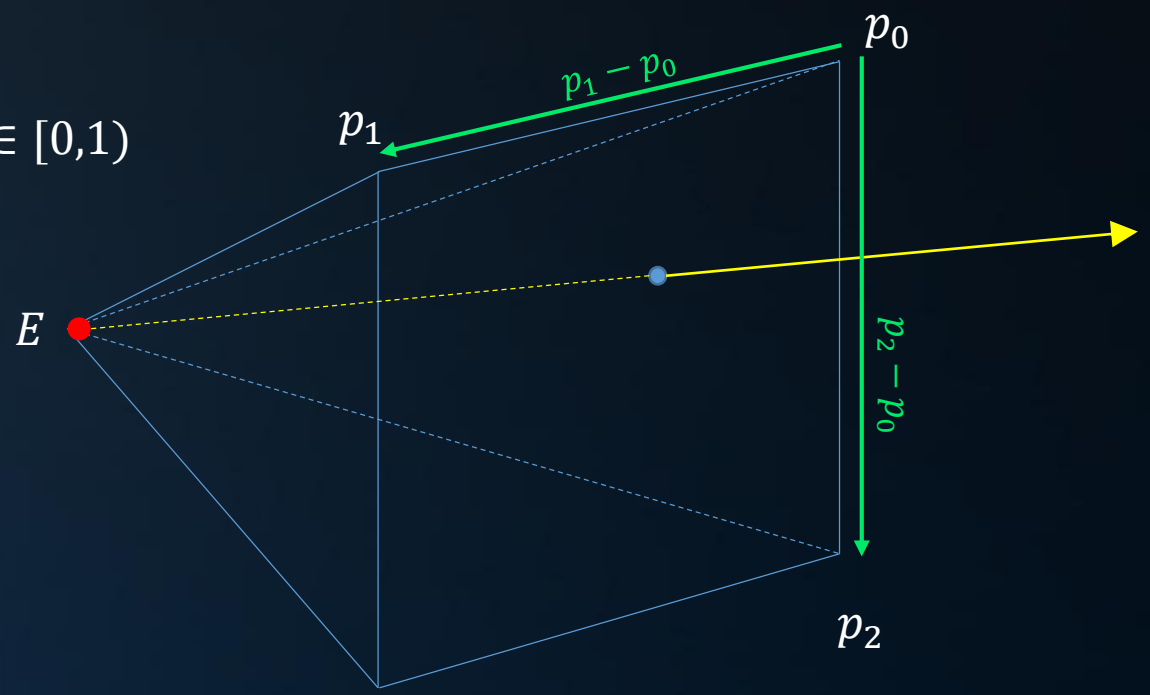
$$p(u,v) = p_0 + u(p_1 - p_0) + v(p_2 - p_0), \quad u,v \in [0,1)$$

Ray direction (before normalization):

$$\vec{D} = p(u,v) - E$$

Ray origin:

$$O = E$$



Intersect

Ray intersection

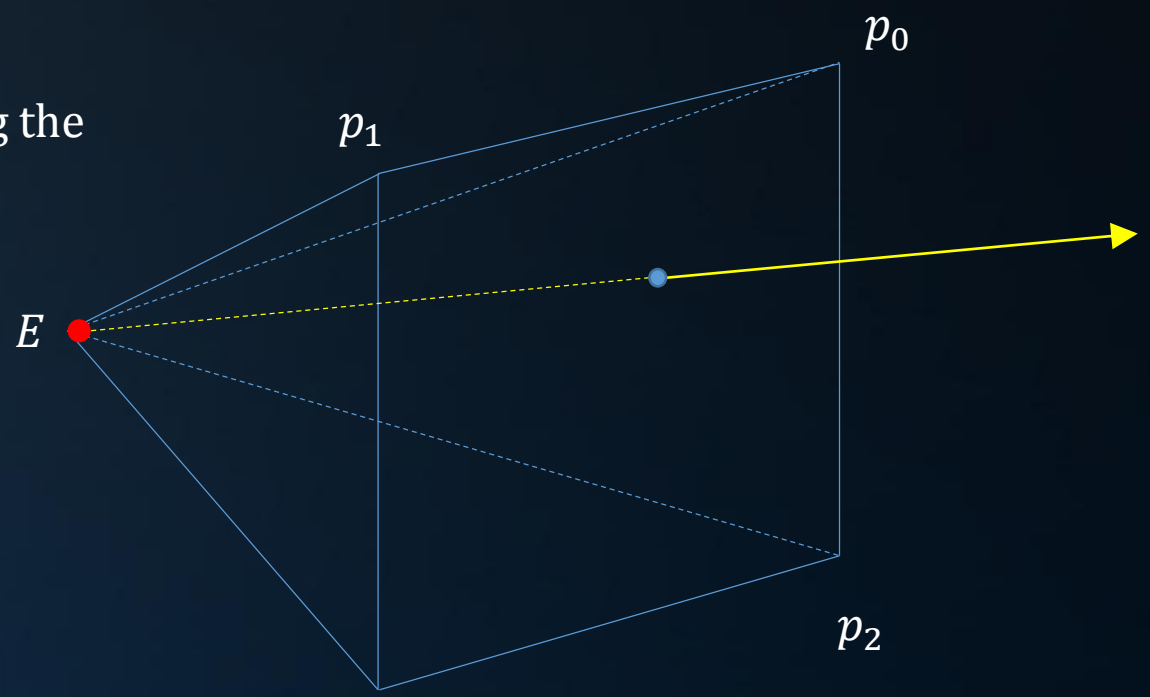
Given a ray $p(t) = O + t\vec{D}$, we determine the smallest intersection distance t by intersecting the ray with each of the primitives in the scene.

Ray / plane intersection:

Plane: $\mathrm{p} \cdot \vec{N} + d = 0$
Ray: $p(t) = O + t\vec{D}$

Substituting for $p(t)$, we get

$$\left(O + t\vec{D}\right) \cdot \vec{N} + d = 0$$
$$t = -(O \cdot \vec{N} + d)/(\vec{D} \cdot \vec{N})$$
$$P = O + t\vec{D}$$



Intersect

Ray intersection

Ray / sphere intersection:

Sphere: $(p - C) \cdot (p - C) - r^2 = 0$
Ray: $p(t) = O + t\vec{D}$

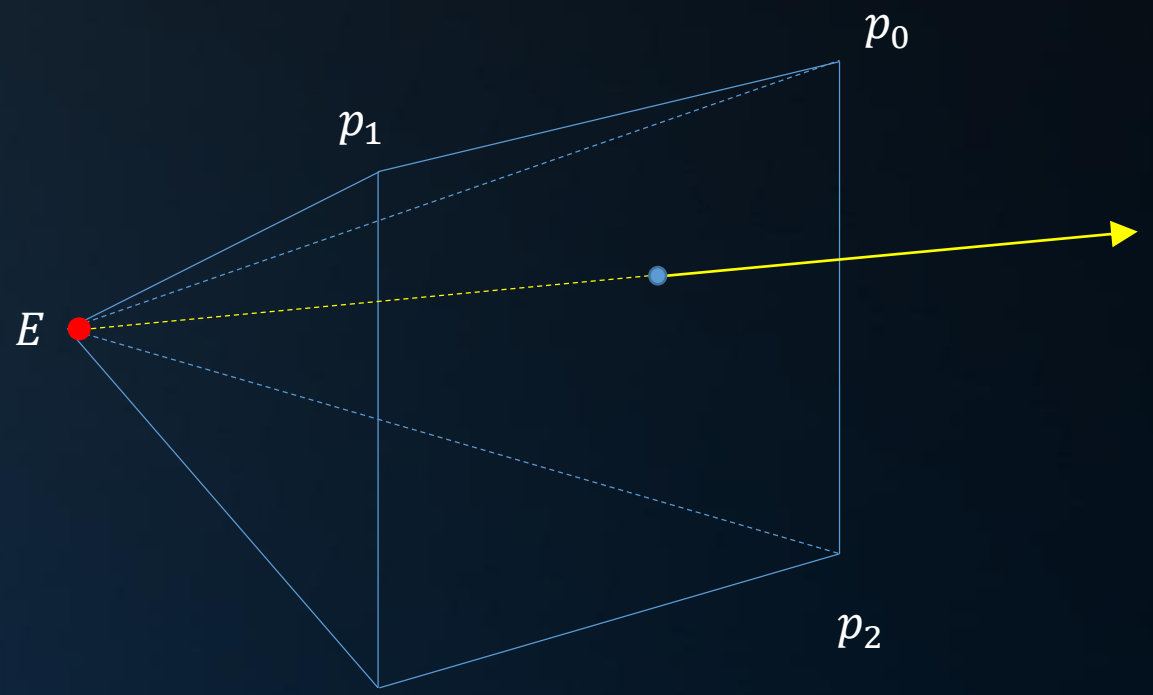
Substituting for $p(t)$, we get

$$\left(O + t\vec{D} - C\right) \cdot \left(O + t\vec{D} - C\right) - r^2 = 0$$
$$\vec{D} \cdot \vec{D}\ t^2 + 2\vec{D} \cdot (O - C)\, t + (O - C)^2 - r^2 = 0$$

$$ax^2 + bx + c = 0 \rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$a = \vec{D} \cdot \vec{D}$$
$$b = 2\vec{D} \cdot (O - C)$$
$$c = (O - C) \cdot (O - C) - r^2$$

Negative:
no intersections

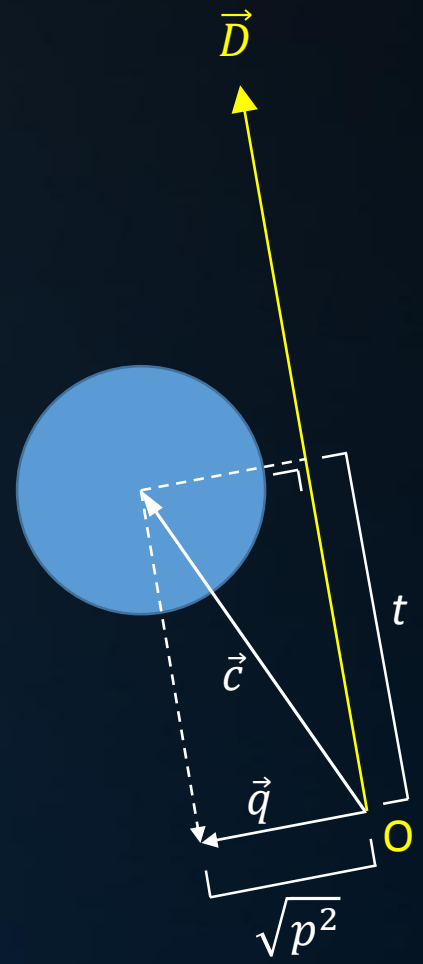


Intersect

Ray Intersection

Efficient ray / sphere intersection:

```
void Sphere::IntersectSphere( Ray ray )
{
    vec3 c = this.pos - ray.O;
    float t = dot( c, ray.D );
    vec3 q = c - t * ray.D;
    float p² = dot( q, q );
    if (p² > sphere.r²) return;
    t -= sqrt( sphere.r² – p² );
    if ((t < ray.t) && (t > 0)) ray.t = t;
    // or: ray.t = min( ray.t, max( 0, t ) );
}
```



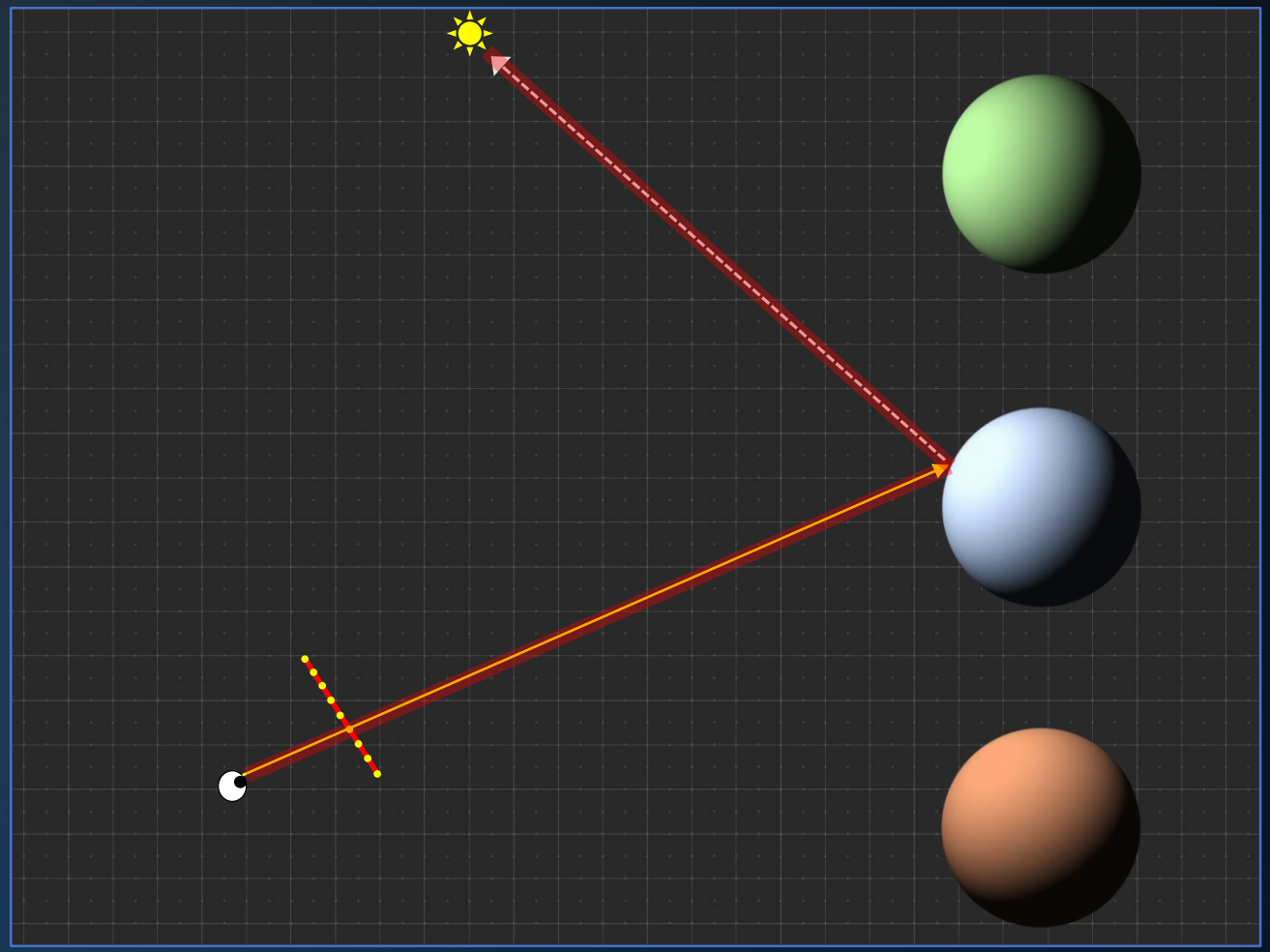
Note:

This only works for rays that start outside the sphere.

Today’s Agenda:

- Ray Tracing
- Intersections
- Shading
- Assignment 2
- Textures

Shading



Shading

The End

We used *primary rays* to find the *primary intersection* point.

Determining light transport:

- Sum illumination from all light sources
- …If they are *visible*.

We used a primary ray to find the object visible through a pixel:
Now we will use a *shadow ray* to determine visibility of a light source.

Shading

Shadow Ray

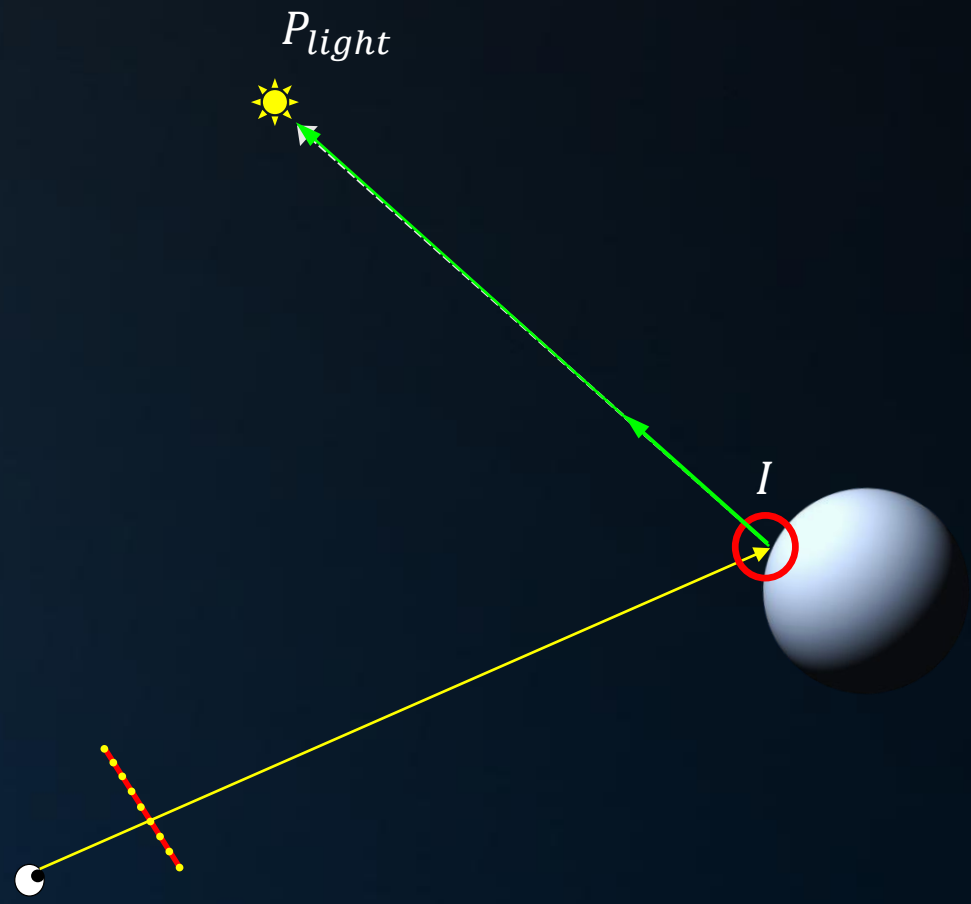
Constructing the shadow ray:

$$p(t) = O + t\vec{D}$$

Ray origin: the primary intersection point I.

Ray direction: $P_{light} - I$ (normalized)

Restrictions on t: $0 < t < ||P_{light} - I||$



Shading

Shadow Ray

Direction of the shadow ray: $\frac{P_{light}-I}{||P_{light}-I||}$

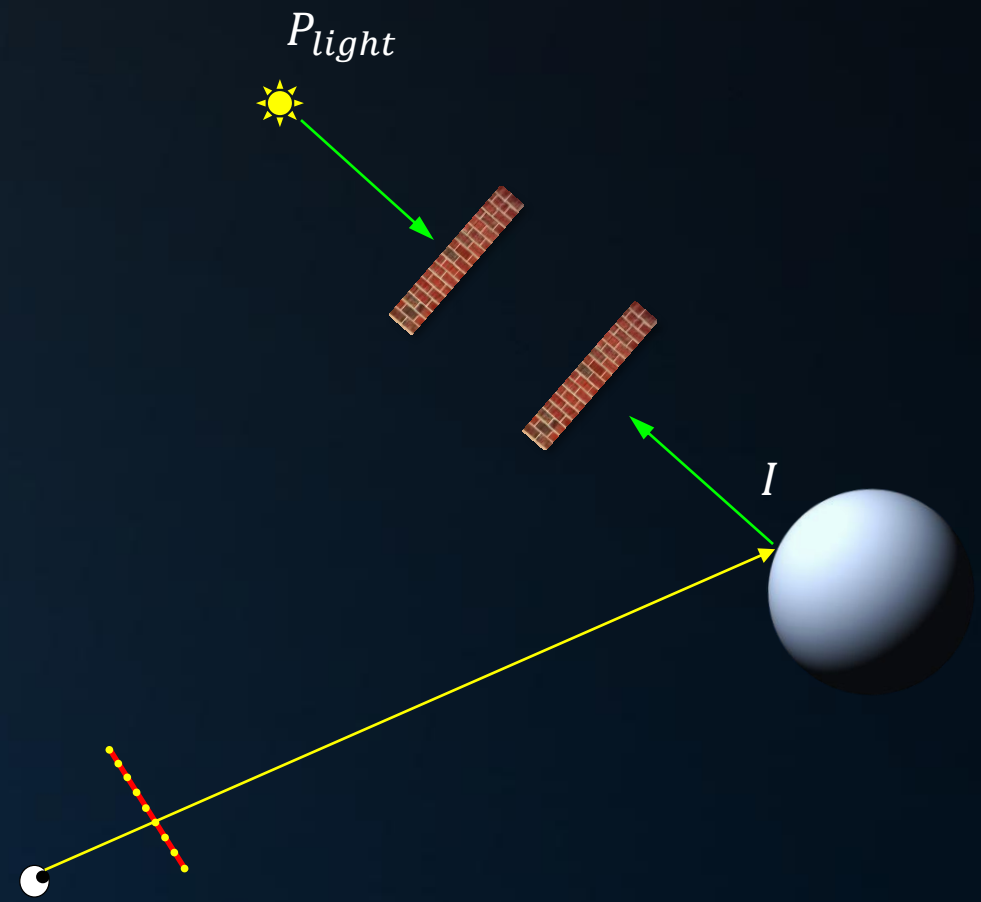
Equally valid: $\frac{I-P_{light}}{||P_{light}-I||}$ or $\frac{I-P_{light}}{||I-P_{light}||}$

Note that we get different intersection points depending on the direction of the shadow ray.

It doesn’t matter: the shadow ray is used to determine *if* there is an occluder, not *where*.

This has two consequences:

1. We need a dedicated shadow ray query;
2. Shadow ray queries are (on average) twice as fast. **(why?)**



Shading

Shadow Ray

“In theory, theory and practice are the same. In practice, they are not.”

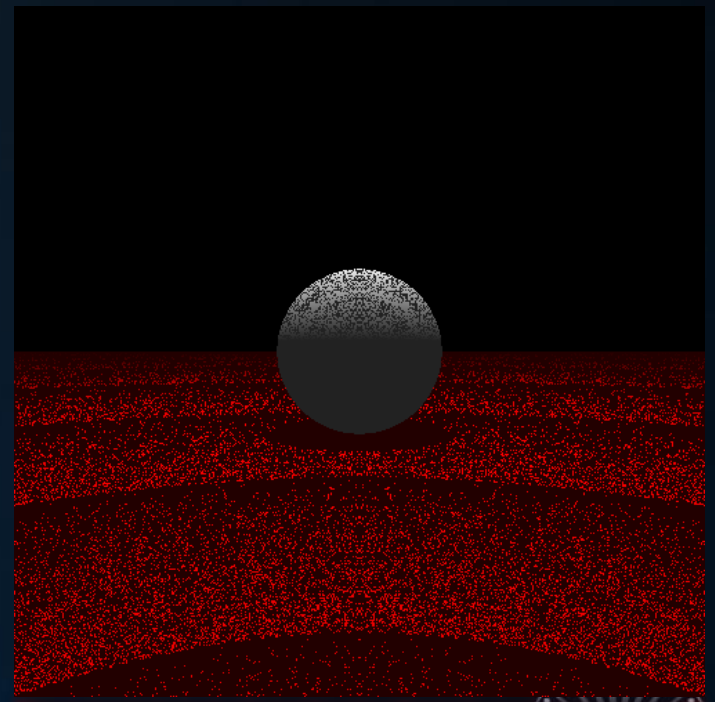
Problem 1:

Our shadow ray queries report intersections at $t = \sim 0$. Why?

Cause: the shadow ray sometimes finds the surface it originated from as an occluder, resulting in *shadow acne*.

Fix: offset the origin by ‘epsilon’ times the shadow ray direction.

Note: don’t forget to reduce t_{max} by epsilon.



Shading

Shadow Ray

“In theory, theory and practice are the same. In practice, they are not.”

Problem 2:

Our shadow ray queries report intersections at $t = t_{max}$. Why?

Cause: when firing shadow rays from the light source, they may find the surface that we are trying to shade.

Fix: reduce $t_{\max}$ by $2 * epsilon$.

Shading

Shadow Ray

“The most expensive shadow rays are those that do not find an intersection.”

Why?

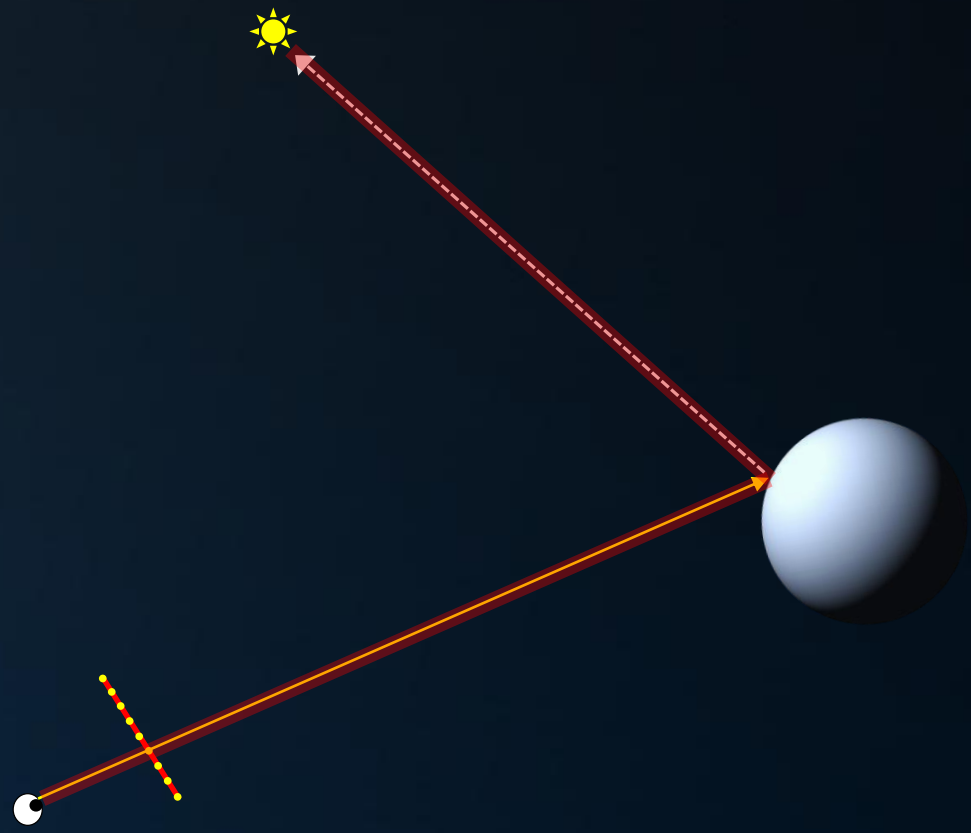
(because those rays tested every primitive before concluding that there was no occlusion)

Shading

Transport

The amount of energy travelling from the light via the surface point to the eye depends on:

- The brightness of the light source
- The distance of the light source to the surface point
- Absorption at the surface point
- The angle of incidence of the light energy



Shading

Transport

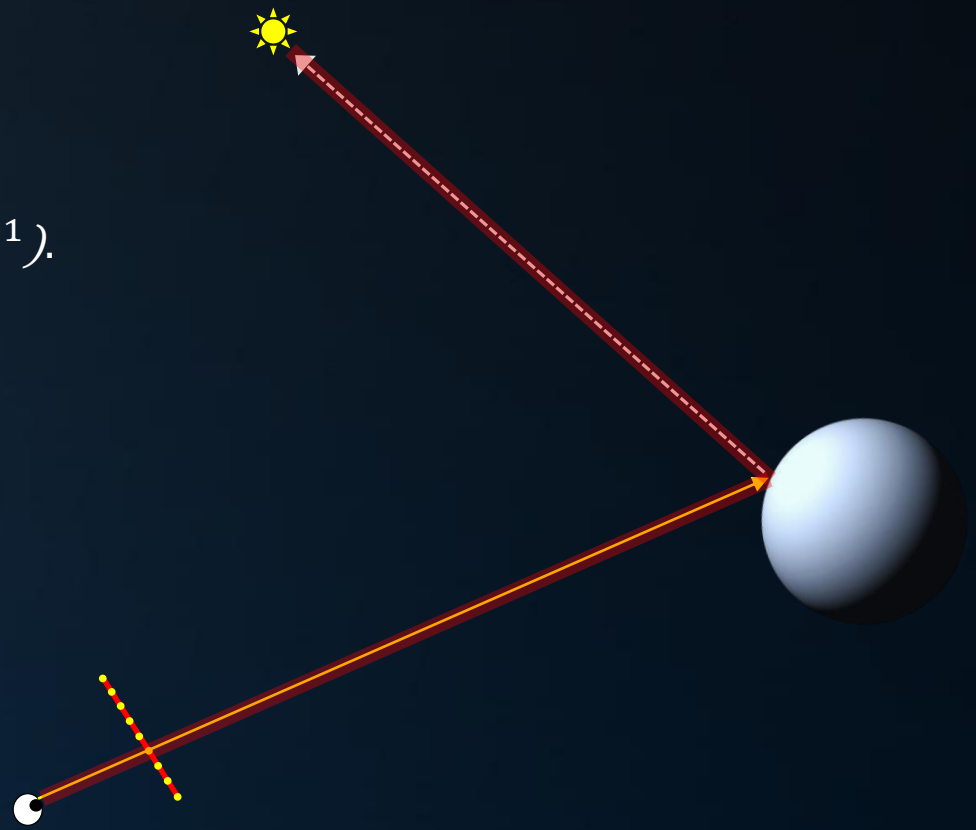
Brightness of the light source:

Expressed in *watt (W)*, or *joule per second* $(J/s \text{ or } Js^{-1})$.

Energy is transported by photons.

Photon energy depends on wavelength; energy for a ‘yellow’ photon is $\sim 3.31 \cdot 10^{-19}$ J.

A 100W light bulb thus emits $\sim 3.0 \cdot 10^{21}$ photons per second.



Shading

Transport

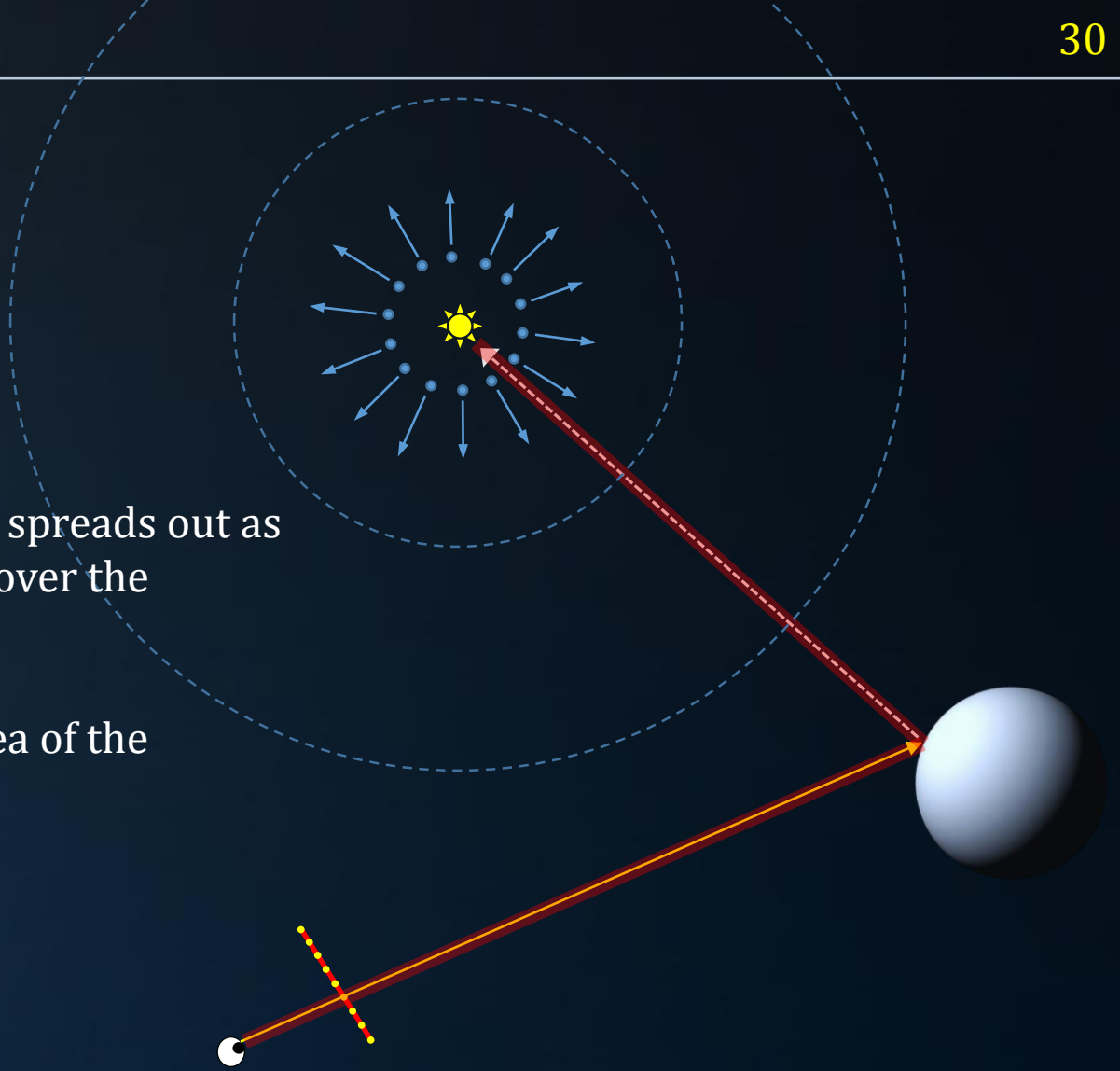
Energy at distance r:

For a point light, a brief pulse of light energy spreads out as a growing sphere. The energy is distributed over the surface of this sphere.

It is therefore proportional to the inverse area of the sphere at distance r, i.e.:

$$E/m^2 = E_{light}\frac{1}{4\pi r^2}$$

Light energy thus dissipates at a rate of $\frac{1}{r^2}$.
This is referred to as *distance attenuation*.



Shading

Transport

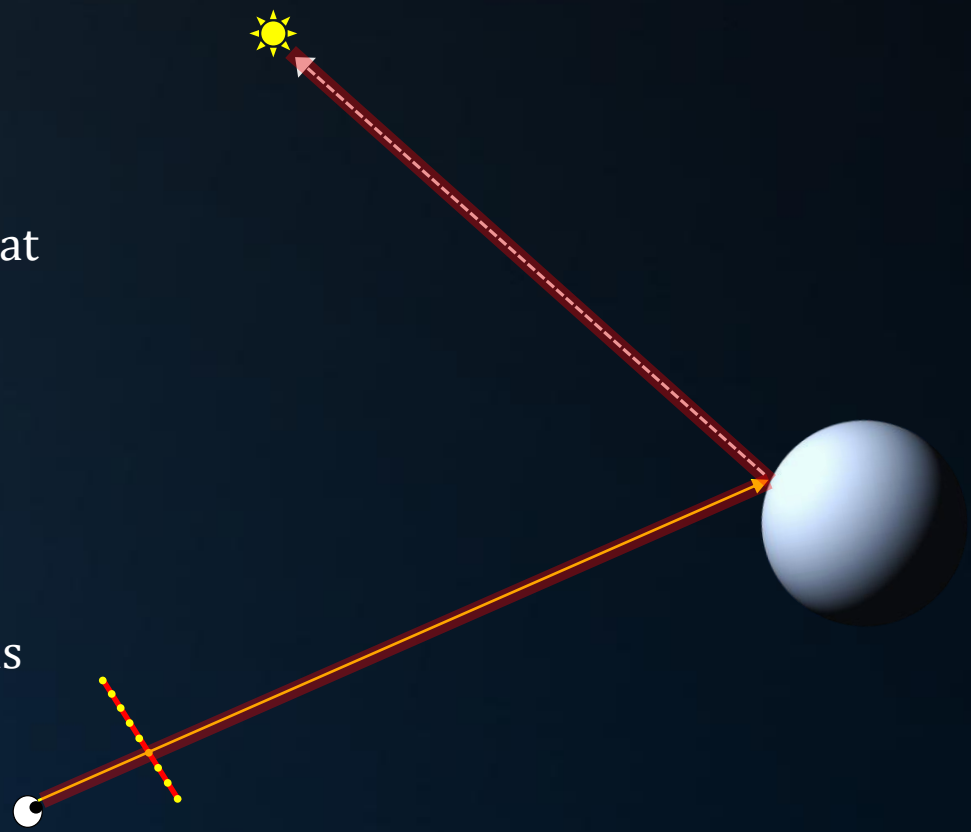
Absorption:

Most materials absorb light energy. The wavelengths that are not fully absorbed define the ‘color’ of a material.

The reflected light is thus:

$$E_{reflected} = E_{incoming} \cdot C_{material}$$

Note that $C_{material}$ cannot exceed 1; the reflected light is never *more* than the incoming light.



Shading

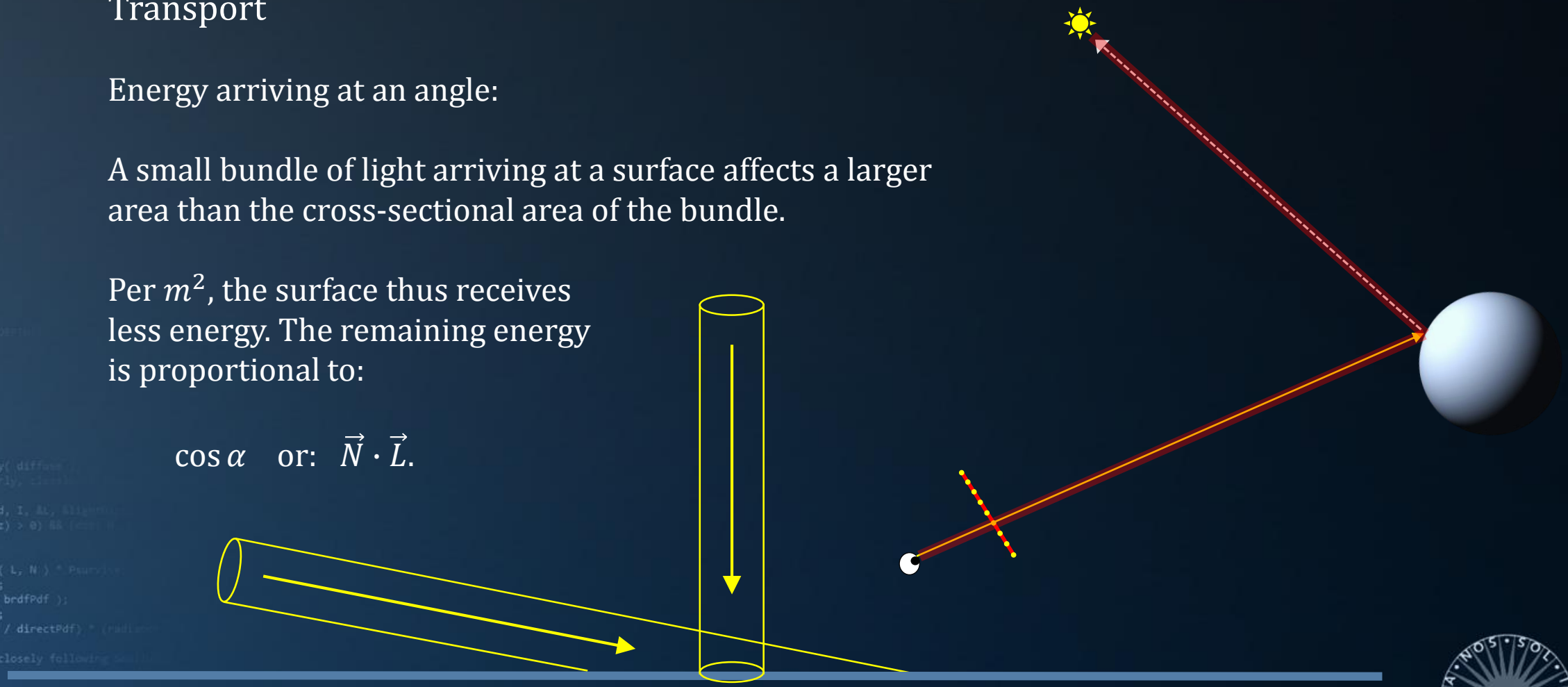
Transport

Energy arriving at an angle:

A small bundle of light arriving at a surface affects a larger area than the cross-sectional area of the bundle.

Per m^2, the surface thus receives less energy. The remaining energy is proportional to:

$\cos\alpha$ or: $\vec{N} \cdot \vec{L}$.

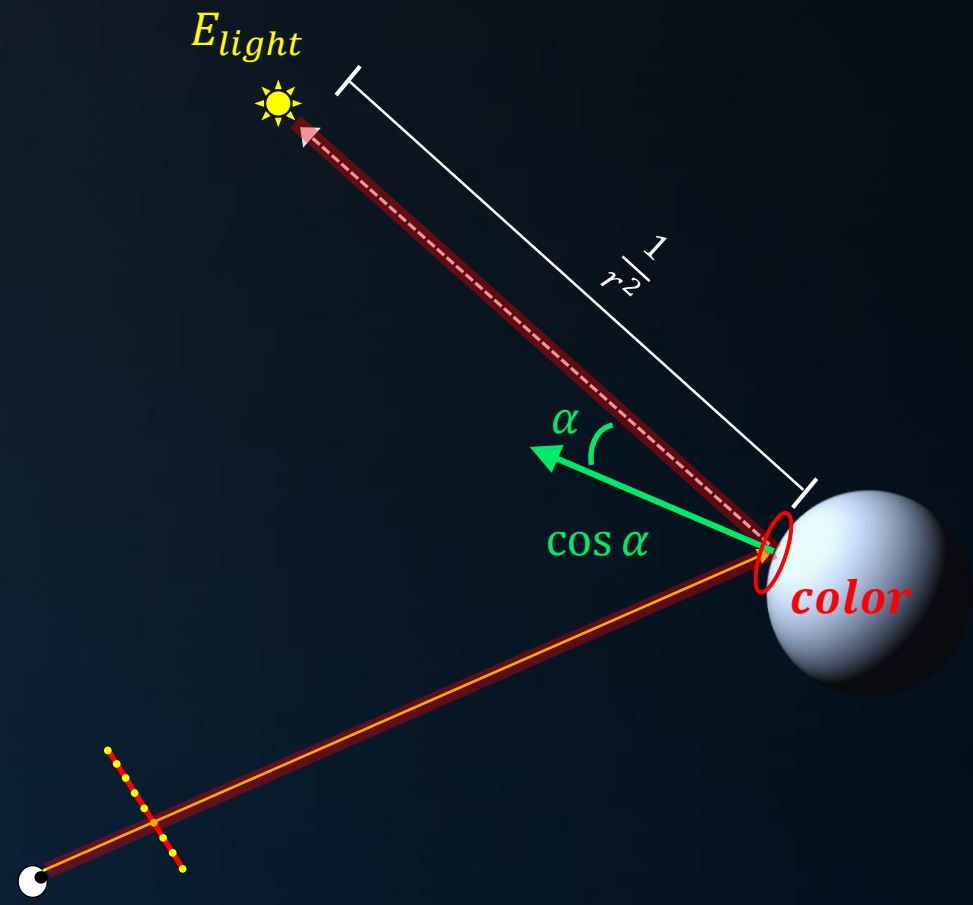


Shading

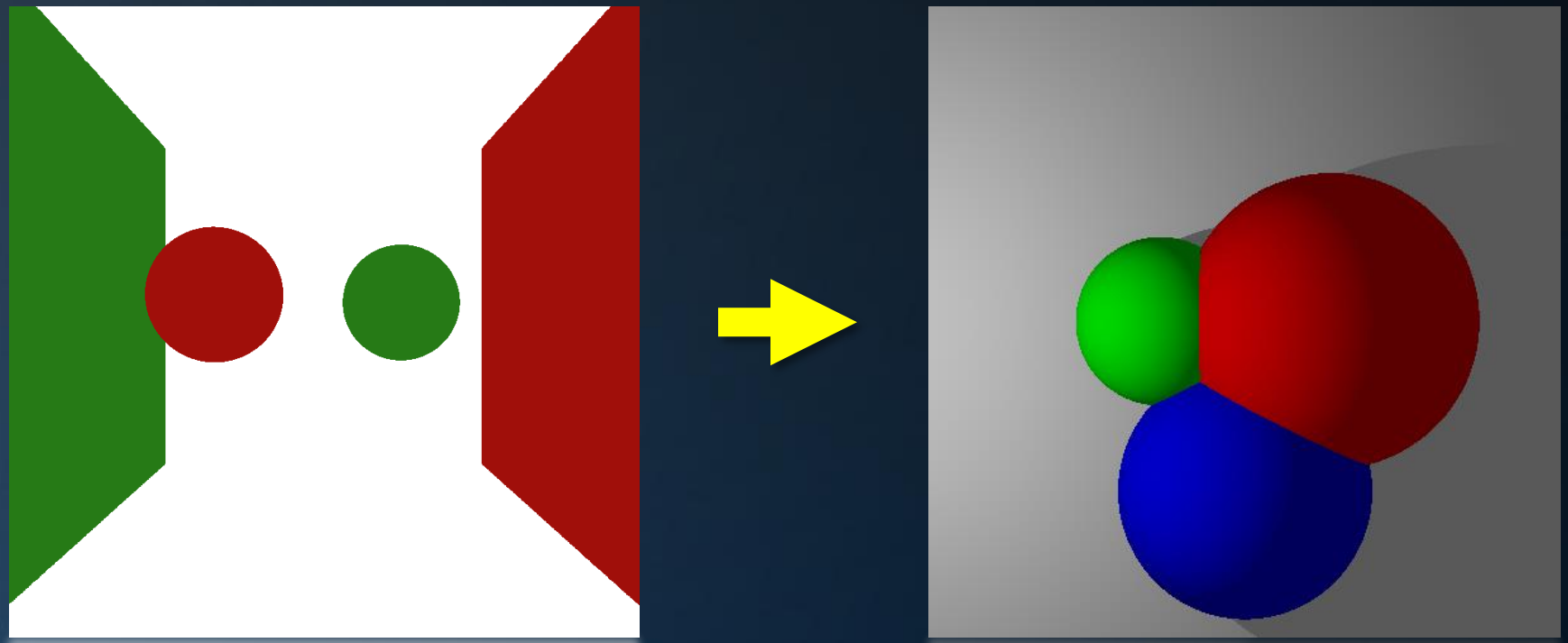
Transport

All factors:

- Emitted light : defined as RGB color, floating point
- Distance attenuation: $\frac{1}{r^2}$
- Absorption, modulate by material color
- N dot L



Shading



Today’s Agenda:

- Ray Tracing
- Intersections
- Shading
- Assignment 2
- Textures

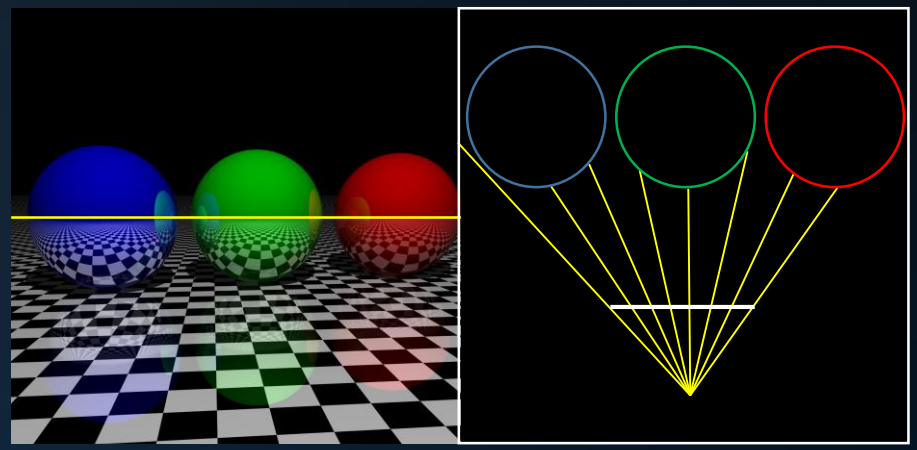
Assignment 2

Deadline assignment 1:

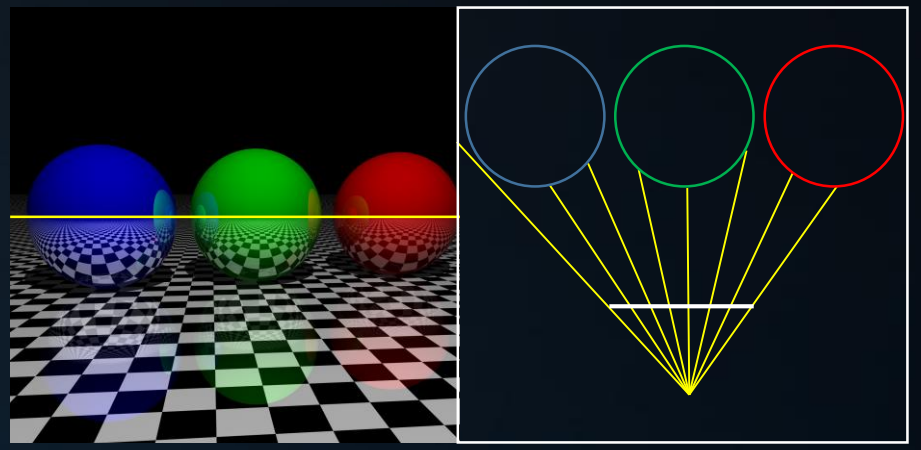
Wednesday May 10th, 23.59

Assignment 2: *”Write a basic ray tracer.”*

- Using the template
- In a 1024x512 window
- Two views, each 512x512
- Left view: 3D
- Right view: 2D slice



Assignment 2



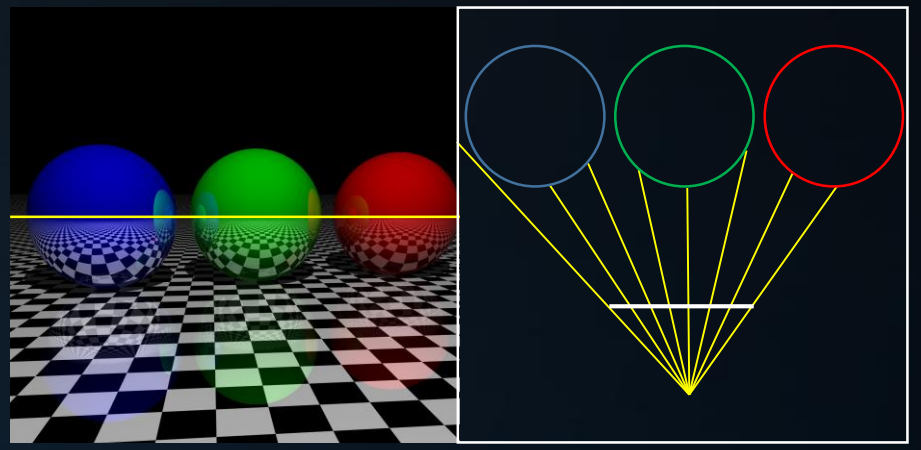
Assignment 2: *”Write a basic ray tracer.”*

Steps:

1. Create a Camera class; default: position (0,0,0), looking at (0,0,-1).
2. Create a Ray class
3. Create a Primitive class and derive from it a Sphere and a Plane class
4. Add code to the Camera class to create a primary ray for each pixel → *For y = 0, visualize every 10th ray*
5. Implement Intersect methods for the primitives
6. Per pixel, find the nearest intersection and plot a pixel → *Visualize the intersection points*
7. Add controls to move and rotate the camera
8. Add a checkerboard pattern to the floor plane.

9. *Add reflections and shadow rays (next lecture).*

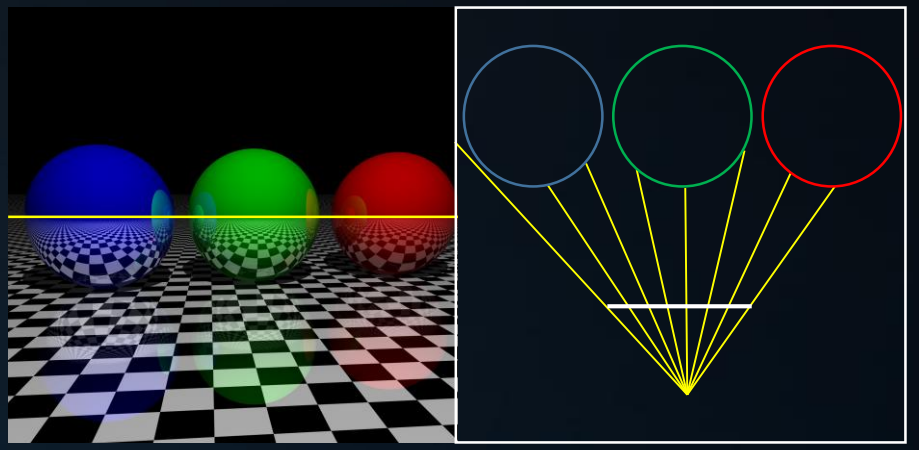
Assignment 2



Extra points:

- Add additional primitives, e.g.:
  - Triangle, quad, box
  - Torus, cylinder
  - Fractal
- Add textures to all primitives
- Add a sky dome
- Add refraction and absorption (next lecture)
- One extra point for the fastest ray tracer
- One extra point for the smallest ray tracer meeting the minimum requirements.

Assignment 2



Official:

- Full details in the official assignment 2 document, available today from the website.
- Deadline: May 30th , 23:59.
- Small exhibition of noteworthy entries in a subsequent lecture and on the website.

Today’s Agenda:

- Ray Tracing
- Intersections
- Shading
- Assignment 2
- Textures

Textures

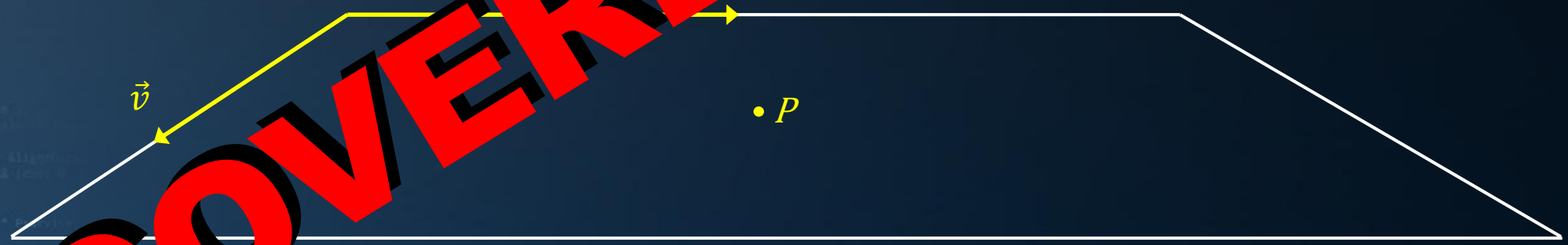
Texturing a Plane

Given a plane: $y = 0$ (i.e., with a normal vector $(0,1,0)$).

Two vectors on the plane define a basis: $\vec{u} = (1,0,0)$ and $\vec{v} = (0,0,1)$.

Using these vectors, a point on the plane can be reached: $P = \lambda_1 \vec{u} + \lambda_2 \vec{v}$.

We can now define a color F: $F(\lambda_1, \lambda_2) = \cdots$.



Today’s Agenda:

- Ray Tracing
- Intersections
- Shading
- Assignment 2
- Textures

INFOGR – Computer Graphics

Jacco Bikker & Debabrata Panja - April-July 2017

END OF lecture 3: “Ray Tracing”

Next lecture: “Ray Tracing (2)”